Accurate Tracking by Overlap Maximization

Goutam Bhat
Abstract

Visual object tracking is one of the fundamental problems in computer vision, with a wide number of practical applications in e.g. robotics, surveillance etc. Given a video sequence and the target bounding box in the first frame, a tracker is required to find the target in all subsequent frames. It is a challenging problem due to the limited training data available. An object tracker is generally evaluated using two criteria, namely robustness and accuracy. Robustness refers to the ability of a tracker to track for long durations, without losing the target. Accuracy, on the other hand, denotes how accurately a tracker can estimate the target bounding box.

Recent years have seen significant improvement in tracking robustness. However, the problem of accurate tracking has seen less attention. Most current state-of-the-art trackers resort to a naive multi-scale search strategy which has fundamental limitations. Thus, in this thesis, we aim to develop a general target estimation component which can be used to determine accurate bounding box for tracking. We will investigate how bounding box estimators used in object detection can be modified to be used for object tracking. The key difference between detection and tracking is that in object detection, the classes to which the objects belong are known. However, in tracking, no prior information is available about the tracked object, other than a single image provided in the first frame. We will thus investigate different architectures to utilize the first frame information to provide target specific bounding box predictions. We will also investigate how the bounding box predictors can be integrated into a state-of-the-art tracking method to obtain robust as well as accurate tracking.
Acknowledgments

Firstly, I would like to thank my examiner Michael Felsberg for giving me an opportunity to work on this interesting thesis. Next, I would like to thank my supervisor Martin Danelljan for his guidance throughout the thesis. I would also like to thank Joakim Johnander and Felix Järemo Lawin for always being available for interesting discussions, and Andreas Robinson for helping with setting up the computers. Finally, I would like to thank everyone at CVL for all the help, and for providing such a wonderful and fun work environment.

Linköping, February 2019
Goutam Bhat
# Contents

1 Introduction ................................................................. 1  
   1.1 Brief Background .................................................. 1  
   1.2 Problem Formulation .............................................. 4  
   1.3 Motivation .......................................................... 5  
   1.4 Thesis Outline ..................................................... 5  

2 Background ...................................................................... 7  
   2.1 Visual Object Tracking .............................................. 7  
      2.1.1 Target Classification .......................................... 8  
      2.1.2 Target Estimation ............................................. 9  
   2.2 Object Detection .................................................... 10  
      2.2.1 IoU-Net ......................................................... 11  

3 Target Classification ..................................................... 13  
   3.1 Classification Model ................................................ 13  
   3.2 Online Learning ...................................................... 14  
   3.3 Implementation Details .......................................... 15  

4 Target Estimation ......................................................... 17  
   4.1 Overview ............................................................. 17  
   4.2 Network Architecture .............................................. 18  
   4.3 Reference Fusion .................................................... 20  
   4.4 Training ............................................................... 21  
   4.5 Online Tracking ...................................................... 22  

5 Experiments ..................................................................... 25  
   5.1 Evaluation Methodology ............................................ 25  
      5.1.1 Evaluation Metrics ........................................... 26  
      5.1.2 Datasets ........................................................ 27  
      5.1.3 Tracker Settings .............................................. 27  
   5.2 Baseline Experiments .............................................. 28  
      5.2.1 Reference Fusion ............................................. 28  
      5.2.2 Backbone features layers .................................. 29
5.2.3 Number of proposals ........................................... 29
5.2.4 Number of iterations .......................................... 30
5.2.5 Multi-Scale search ............................................ 30
5.3 State-of-the-art Evaluation .................................. 31
  5.3.1 Need For Speed ............................................. 31
  5.3.2 UAV123 ......................................................... 31
  5.3.3 TrackingNet .................................................. 32
  5.3.4 VOT2018 ....................................................... 32
5.4 Attribute-based Comparison .................................. 33
5.5 Qualitative Results ............................................... 33

6 Conclusion .......................................................... 37

Bibliography .......................................................... 39
Generic visual object tracking refers to the problem of estimating the trajectory of an object in a video sequence, given its location in the first frame. It is one of the fundamental problems in computer vision and can serve as a building block in complex vision systems, e.g. automatic surveillance, autonomous driving. This thesis will deal with the problem of improving the accuracy of state-of-the-art trackers. The rest of the chapter is organized as follows. Section 1.1 provides a brief background to the problem of object tracking and target state estimation. Section 1.2 formulates the problems tackled in the thesis. Section 1.3 gives a motivation to these problems. Section 1.4 contains an outline for the rest of the thesis.

1.1 Brief Background

Visual object tracking is one of the fundamental problems in computer vision, with a wide number of practical applications. Given a video sequence and the target state in the first frame (generally represented as an axis-aligned bounding box), a tracker is required to find the target states in all subsequent frames. It is a challenging problem due to a number of reasons. Firstly, a tracker is required to track objects belonging to any arbitrary class. This is in contrast to other computer vision fields such as classification and detection where the objects are assumed to belong to a known set of classes. Secondly, the tracked object can undergo significant appearance changes due to, e.g. deformation, motion blur, occlusion, illumination variation. Further, the scene can contain other distractor objects which look similar to the target (see figure 1.1 for an illustration). Thus, the tracker must learn a robust appearance model of the target which can handle these challenges from a single image as training data.

An object tracker is generally evaluated using two criterias, namely robustness and accuracy. Robustness refers to the ability of a tracker to track for long durations, without losing the target. In order to achieve high robustness, a tracker must be able to reliably
Introduction

Figure 1.1: Example sequences from standard tracking datasets. The tracker is provided the target annotation for the first frame, and is required to determine target bounding box for all subsequent frames. The target can undergo significant appearance changes due to e.g. rotation, deformation, motion blur, occlusions. Further, the scene can contain background distractors. This makes the tracking problem extremely challenging.
distinguish the target from the background and other distractor objects. Accuracy, on the other hand, denotes how accurately a tracker can estimate the target state. A tracker must be able to precisely determine the object boundary in the image to achieve high accuracy.

The tracking problem can be decomposed into two sub-tasks, one of which caters to robustness, and another provides accuracy. In the first task, named target classification, the aim is to reliably provide a coarse location of the target in the image by classifying image patches into foreground and background. In order to perform robust target classification, a tracker is required to learn an appearance model of the target invariant to changes in the target appearance caused by, e.g., deformation, motion blur, illumination variation. Further, the tracker must also be able to distinguish the target object from background distractors, i.e., background regions with appearance similar to the target.

The second sub-task, target estimation, is then to precisely estimate the target state, often represented by a bounding box, in order to achieve high tracking accuracy. This is a challenging problem since the tracker should have high level knowledge about the target object in order to determine its state. The target can undergo deformations, or can have out-of-plane rotations. In these cases, the tracker must know ‘what’ constitutes an object boundary to determine the bounding-box coordinates. The problem is complicated by the fact that the tracker is provided only one ‘view’ of the target as training data.

In recent years, the focus of tracking research has been on target classification. Much attention has been invested into constructing robust classifiers, based on, e.g., correlation filters [6, 24, 34], and exploiting powerful deep feature representations [3, 37] for this task. On the other hand, target estimation has seen below expected progress. This trend is clearly observed in the recent VOT2018 challenge [19], where older trackers such as KCF [15] and MEEM [44] still obtain competitive accuracy while exhibiting vastly inferior robustness. In fact, most current state-of-the-art trackers [3, 8, 34] still rely on the classification component for target estimation by performing a multi-scale search. However, this strategy is fundamentally limited since bounding box estimation is inherently a challenging task, requiring high-level understanding of the object’s pose and thus cannot be modeled by simple image transformations, such as scaling (see figure 1.2 for an illustration). This highlights the need for a separate target estimation component which can handle changes in target state brought about by, e.g., deformation, rotations.

Another computer vision problem which has many similarities with tracking is object detection. Here, the problem is to locate all the instances of objects belonging to a certain number of classes and determine which class an object belongs to. Recent object detectors generate a set of candidate target locations using either object proposals [11] or anchor boxes [30]. For each of these candidates, one component determines the object class, while another component predicts the bounding box co-ordinates for the object, given a rough initial estimate. Bounding box prediction is commonly performed using a technique called bounding box regression, where a network directly predicts the offsets to be applied to the initial estimate in order to get the final bounding box. Recently, another approach based on overlap prediction has been proposed and has been shown to provide more accurate bounding boxes [16]. Here, a network predicts the intersection-over-union (IoU) overlap between the target and an estimated bounding box. The key idea is that the IoU prediction is differentiable w.r.t. the bounding box co-ordinates. Thus, the final bounding box prediction can be obtained by maximizing the IoU through gradient ascent.
1 Introduction

Figure 1.2: Tracking results from a state-of-the-art tracker, UPDT [3]. UPDT employs a brute-force multi-scale search for target state estimation. While UPDT can achieve robust tracking in all the cases, it fails to handle the changes in target state due to e.g. rotation, camera motion, and thus cannot provide accurate bounding boxes. Specially, observe that UPDT cannot handle changes in target aspect ratio and is constrained to provide boxes with fixed aspect ratio.

1.2 Problem Formulation

The goal of this thesis is to develop a general target estimation component which can be used along with any classifier to achieve accurate tracking. As discussed before, the target estimation task requires high-level understanding of the object’s pose which cannot be learnt from a single image. Thus, the target estimation component should be trained offline, using large-scale video datasets in order to learn a general representation useful for target estimation. However, it should still use the training data from the first frame in order to make accurate predictions for the specific object which is being tracked.

In order to develop a target estimation component, this thesis will investigate how the bounding box prediction techniques used in object detection can be modified to be used in object tracking. We will focus on the overlap maximization approach [16] which has been shown to provide accurate bounding boxes. For the task of object detection, independent bounding box estimators are often trained for each object class. However, in tracking, the class of the tracked object is generally unknown. Further, the target object is not required to belong to any set of pre-defined classes or be represented in any existing training datasets. Class-specific bounding box estimators are thus of little use for generic visual tracking. Instead, target-specific estimators, which can exploit the target annotation in the
first frame need to be developed. The main challenge here is thus how to integrate the first frame information in the bounding box estimator. We will investigate this problem in the thesis. Further, we will study the problem of integrating the target estimation component with a target classifier in order to achieve robust as well as accurate tracking.

1.3 Motivation

Visual object tracking is a fundamental problem in computer vision with several applications. Object tracking can be used in autonomous driving systems to keep track of other vehicles or pedestrians. Another common application is surveillance systems, where a tracker can be used to determine how a person is moving inside an e.g. airport, train station.

While having high accuracy tracking might not be necessary for all tracking applications, it is crucial in many applications. E.g. if a tracker is being used to follow a person from an Unmanned Aerial Vehicle (UAV), then the bounding box estimated by the tracker can be used to calculate the distance between the person and the UAV. This estimated distance can then be used by the UAV to maintain a safe distance from the person. Any error in bounding box estimation might thus lead to the UAV flying too close to the person, which is not desirable. Similarly, a tracker can be used in a goal-line system to determine if the football has crossed the goal-line. Accurately estimating the bounding box coordinates of the ball is thus important to make the correct decision.

Even in applications where high accuracy is not needed, accurate tracking can still be useful to indirectly provide high robustness. Consider the scenario where an object is moving away from the camera, thus shrinking in size. If the target estimation is inaccurate, resulting in a bounding box much larger than the actual target, then the classifier will be evaluated at an incorrect scale. That is, the tracker will try to find an object of a totally different size, which might result in tracking loss. Further, many trackers update the target classifier using the previously tracked frames, with the tracking output as groundtruth. In such cases, inaccurate bounding box estimation might lead to corruption of the target classifier due to noisy annotation. This will ultimately lead to tracking loss.

1.4 Thesis Outline

The rest of the thesis is organized as follows. Chapter 2 provides a background to the field of object tracking, as well as the overlap maximization approach for bounding box prediction for object detection. Chapter 3 describes the target classification module used for the thesis. The target estimation component developed in this thesis is described in chapter 4. Chapter 5 provides experimental evaluations and results on standard tracking benchmarks. The conclusions from the thesis are presented in chapter 6.
The goal of this thesis is to develop a general target estimation component which can accurately determine the bounding box coordinates of the object being tracked. The thesis will specifically investigate how the recent advancements in object detection can be used to develop a target estimator for visual tracker. This chapter provides a brief background to the fields of visual tracking (Section 2.1) and overlap maximization technique from object detection (Section 2.2).

2.1 Visual Object Tracking

Visual object tracking is a classical computer vision problem which has been studied extensively. The aim in tracking is to determine the trajectory of an object in a video sequence, given its location in the first frame. It is a challenging problem due to the limited amount of training data available (a single image). The tracked object can undergo significant appearance changes due to, e.g. deformation, motion blur, occlusion, illumination variation. Thus, the tracker must learn a robust appearance model of the target which can handle these difficulties from a single image as training data.

For successful tracking, a tracker must first be able to distinguish the target object from the background and other objects in the scene. Secondly, it must also be able to estimate the state of the target in every frame, that is, find the exact target center and size in the image. Both these tasks are quite challenging in different ways. For the first task, which we will refer to as target classification, the tracker must learn an appearance model of the target which can discriminate the target from the background, and is invariant to changes in target appearance caused by e.g. rotations, deformation, motion blur. Section 2.1.1 contains strategies used by different trackers for the target classification tasks. In case of the second task, called target estimation, the tracker needs to know what constitutes the object boundary so that it can determine a bounding box which best fits the object. Section 2.1.2 describes how target estimation is generally performed. Both the target
2.1 Target Classification

In recent years, several approaches have been proposed to tackle the problem of target classification in tracking. One of the approaches which has gained significant attention and attained state-of-the-art results is Discriminative Correlation Filters (DCF) [4, 8, 17]. As the name suggests, these methods discriminatively train a correlation filter, which models the appearance of the target, online, i.e. in the first frame. The appearance model is trained using the image patches containing the target as positive training samples and the background regions as negative training samples. The default strategy is to use all the shifted versions of the image for training as that allows the problem to be diagonalized and solved efficiently in the Fourier domain.

The early methods used simple training procedures and could operate at high frame rates (>100 FPS). Subsequently, a number of improvements have been proposed to robustify the classifier. SRDCF [6] introduces a spatial regularization term in the classifier learning which penalizes the filter coefficients based on their spatial location. This enables the classifier to be trained using a larger image region, using more training data. BACF [17] also proposes an approach to utilize more training data by sampling negative examples from the full image, in contrast to standard DCF which only uses circularly shifted patches around the target. CSRDCF [24] learns spatial and channel reliability which helps the filter to focus on object region best suitable for tracking. CCOT [7] trains a filter in the continuous domain which allows simultaneous use of features with different resolutions, while also providing sub-pixel accuracy. Its descendent, ECO [8] introduces a factorized convolution operator for dimensionality reduction and a generative model of the training data which provide a drastic increase in tracking speed while reducing classification and target estimation tasks are illustrated in figure 2.1.

2.1.1 Target Classification

Figure 2.1: An illustration of the target classification (a) and target estimation (b) tasks. Given an image, the goal in target classification is to determine a coarse location of the target (red box). The target estimator should then refine this box to provide a more accurate bounding box (green box) for the target.
Along with the development of robust learning methods, another factor which has led to the improvements in DCFs is the use of powerful features. Initially, single channel grayscale intensity values were used as input features for the appearance model [4]. Later, hand-crafted features such as histogram-of-gradients (HOG) [5] and color names (CN) [38] were employed. The recent trend however, is to use the features extracted from deep layers of a convolutional neural network (CNN) [7, 25, 34]. These high-dimensional deep features exhibit superior invariance to appearance changes in the target caused by e.g. rotation, deformation compared to the hand-crafted features, and thus, provide much higher robustness. The most common strategy is to use features from networks trained for image classification on the ImageNet [32] dataset. However, several attempts have been made to train networks specifically for tracking [37, 42].

Another family of target classifiers which has become popular in recent years, specially for high speed tracking, is Siamese methods [2, 20, 35]. These methods train a network to predict the similarity score between two images of the same size. Images of the same object from different frames in a sequence must be given high similarity score by the network, while the images of different objects should be given low scores. This network can be trained offline using large video datasets. During online tracking, the target is found in each frame by finding the image region most similar to the target region in the first frame. Thus, no appearance model needs to be trained online, leading to high tracking speeds. However, this also leads to lower resilience to distractors. RASNet [39] attempts to solve this by predicting residual attentions to select the spatial regions and channels which are most discriminative to track the current target. SiamRPN [20] uses a region proposal network (RPN) [31], commonly used in object detection, on top of a siamese network. The RPN classifies each image patch into target or background. The filter weights for the RPN are predicted by template branch of the siamese network, using the first frame. DaSiamRPN [45] further improves SiamRPN by performing distractor aware training, i.e. explicitly training the network to handle distractor objects.

In addition to DCFs and Siamese methods, several other deep learning based approaches have also been proposed for target classification. MDNet [28] trains a Convolutional Neural Network (CNN) binary classifier for tracking. The network consists of several shared layers and domain specific layers. The shared layers trained offline, using large video datasets to learn general representation useful for target classification. The domain specific layers on the other hand are trained for a particular sequence. Tracking is performed by sampling a few regions around the previous known location of the target and selecting the region with highest score. GOTURN [14] directly regresses the target bounding box given the current frame, and a patch centered at the target from the previous frame. ADNet [43] uses deep reinforcement learning to train a network which predicts the actions to be performed on the previous bounding box estimate to fit the target in the current frame. The actions can be shifting of the box (left, right, up, bottom) or changing the size of the box (expand, shrink).

### 2.1.2 Target Estimation

In contrast to target classification, target estimation has received much lesser attention in the recent years. Most current state-of-the-art trackers [3, 8, 34] do not use a separate com-
ponent for target estimation. Instead, they rely on the classification component for target estimation by performing a multi-scale search. That is, the image is resized to several different scales and the classifier applied on each of them. The location and the scale which gives the highest score is then selected as target state. There are two major drawbacks to this approach. Firstly, it only models simple scaling of the target. Aspect ratio changes in the target, brought about by pose change cannot be handled by this approach. Secondly, the classifier is trained to distinguish the target from the background and not for finding target state. This might result in less accurate state estimation.

A target estimation approach which is widely used is the discriminative scale space tracker (DSST). This approach trains an explicit filter online for scale estimation, by sampling the target at different scales. However, the training data only contains scaled versions of the target, and not changes in target size caused by deformation or rotations. Further, as in multi-scale search, this approach is also unable to handle the aspect ratio changes in the target.

A few approaches have utilized bounding box regression techniques commonly used in object detection. MDNet [28] trains a bounding box regressor to predict the bounding box coordinates, using the groundtruth annotation in the first frame. Unlike DSST, this approach can predict boxes with different aspect ratios. However, since the regressor is trained on a single image, it suffers from the lack of diverse training data. SiamRPN [20], as discussed before, uses a region proposal network (RPN), on top of a siamese network. One branch of the RPN classifies each anchor box into target or background, while another branch predicts the offsets to be added to the anchor box to get the bounding box for the target. The network is trained offline end-to-end using large scale video datasets. Thus it can learn a general representation for various objects, and thus better handle the changes in target state due to pose changes. One drawback of this approach is that target estimation is closely coupled with target classification, since both are performed by the RPN. Thus, it cannot be directly used with another target classifier.

2.2 Object Detection

Given an image, the aim in object detection is to detect all instances of objects belonging to a certain set of classes, say human, car, cat and dog. By ‘detect’, we mean that for every object in the image, the object detector should determine its class and the location in the image, generally represented as a bounding box. Note that this is somewhat similar to visual tracking, the major difference being that in tracking, the aim is to detect one specific object belonging to any arbitrary class, instead of objects belonging to certain classes. Due to this similarity, the advances in object detection have been incorporated for tracking in several instances [20, 28].

For this thesis, we are interested in bounding box estimation techniques used for object detection. One of the common strategy in object detection is to use a region proposal network [31] to obtain a number of object proposals, that is, regions which contain an object. Next, the features from the regions corresponding to each of the proposal are pooled to a fixed length vector using a region-of-interest (ROI) pooling layer. These pooled features are then passed through a classification branch, which determines the object class for that proposal. Simultaneously, the features are passed through another
Figure 2.2: An illustration of the standard object detection pipeline. Given an image, a region proposal network (not shown here) is used to obtain a number of bounding box proposals for the objects. Next, the features from the regions corresponding to each of the proposal are pooled to a fixed length vector using a region-of-interest (ROI) pooling layer. These pooled features are then passed through a classification branch, which determines the object class, and a bounding box estimator branch, which refines the proposal bounding box.

The common strategy is to use a bounding box regressor which predicts the offsets to the proposal in order to get the bounding box for the object [11, 30]. Recently, another approach based on IoU prediction [16] has been proposed for bounding box estimation. This approach, named IoU-Net, has been shown to provide more accurate bounding boxes, compared to the standard bounding box regression approach. We briefly describe IoU-Net in the next section.

2.2.1 IoU-Net

IoU-Net [16], was recently proposed for object detection as an alternative to typical anchor-based bounding box regression techniques. In contrast to conventional approaches, the IoU-Net is trained to predict the IoU between an image object and an input bounding box candidate. Bounding box estimation is then performed by maximizing the IoU prediction.

Given a deep feature representation of an image, $x \in \mathbb{R}^{W \times H \times D}$, and a bounding box estimate $B \in \mathbb{R}^4$ of an image object, IoU-Net predicts the IoU between $B$ and the object. Here $B$ is parametrized as $B = (c_x/w, c_y/h, \log w, \log h)$, where $(c_x, c_y)$ are the image coordinates of the bounding box center. The network uses a Precise ROI
Figure 2.3: An illustration of the IoU-Net architecture. IoU-Net uses Precise ROI pooling layer to pool the features corresponding to the proposal region to a fixed length vector. This vector is then passed through a fully connected layer which predicts the IoU overlap between the proposal box and the object. The proposal is then refined by maximizing the predicted IoU w.r.t. the proposal co-ordinates using gradient ascent.

Pooling \( \text{PrPool} \) [16] layer to pool the region in \( x \) given by \( B \), resulting in a feature map \( x_B \) of a pre-determined size \( R \times C \). \text{PrPool} interpolates the input feature map to a continuous domain using bilinear interpolation. The interpolated feature map is then divided into uniform grid cells of size \( R \times C \). The values within each feature map are integrated to get the output feature map. Compared to other pooling approaches such as max pool or average pool, the key advantage of \text{PrPool} is that the output is differentiable w.r.t. the bounding box coordinates \( B \). This allows the bounding box \( B \) to be refined by maximizing the IoU w.r.t. \( B \) through gradient ascent. See figure 2.3 for an illustration.
In this chapter, we describe the target classification component employed in our method. As the focus of the thesis is on developing a general target estimation component, we will use a rather simple target classifier.

### 3.1 Classification Model

Our target classification module is a 2-layer fully convolutional neural network, formally defined as

$$f(x; w) = \phi_2(w_2 \ast \phi_1(w_1 \ast x)). \quad (3.1)$$

Here, $x$ is the backbone feature map which can either be obtained using hand-crafted methods such as histogram of gradients (HOG) [5], or as output of some pre-trained deep neural network. The weights $w = \{w_1, w_2\}$ are the parameters of the classification network and are learned online for each target. $\phi_1, \phi_2$ are activation functions (e.g. Sigmoid, ReLU) which can be used to introduce non-linearity in the model. The operation $\ast$ denotes standard multi-channel convolution. While the framework is general and can be extended with more layers or complex network architectures, we found such a simple model sufficient and beneficial in terms of computational efficiency.

The parameters of the model, $w$, are learned using the objective based on the $L^2$ classification error which is commonly used in DCF based tracking approaches,

$$L(w) = \sum_{j=1}^{m} \gamma_j \|f(x_j; w) - y_j\|^2 + \sum_k \lambda_k \|w_k\|^2. \quad (3.2)$$

Each training sample feature map $x_j$ is annotated by the classification confidences $y_j \in \mathbb{R}^{W \times H}$, set to a sampled Gaussian function centered at the target location. The impact of each training sample is controlled by the weight $\gamma_j$, while the amount of regularization on $w_k$ is set by $\lambda_k$. 

3.2 Online Learning

A brute-force approach to minimize (3.2) would be to apply standard gradient descent or its stochastic twin. These approaches are easily implemented in modern deep learning libraries, but are not well suited for online learning due to their slow convergence rates. We therefore use a more sophisticated optimization strategy that is tailored for such online learning problems, yet requires only little added implementation complexity. First, we define the residuals of the problem as \( r_j(w) = \sqrt{\gamma_j}(f(x_j; w) - y_j) \) for \( j \in \{1, \ldots, m \} \) and \( r_{m+k}(w) = \sqrt{\lambda_k} w_k \) for \( k = 1, 2 \). The loss (3.2) is then equivalently written as the squared \( L^2 \) norm of the residual vector \( L(w) = \|r(w)\|^2 \), where \( r(w) \) is the concatenation of all residuals \( r_j(w) \).

We utilize the quadratic Gauss-Newton approximation \( L_w(\Delta w) \approx L(w + \Delta w) \), obtained from a first order Taylor expansion of the residuals \( r(w + \Delta w) \approx r_w + J_w \Delta w \) at the current estimate \( w \).

Here, we have defined \( r_w = r(w) \) and \( J_w = \frac{\partial r}{\partial w} \) is the Jacobian of \( r \) at \( w \). The new variable \( \Delta w \) represents the increment in the parameters \( w \).

The Gauss-Newton subproblem (3.3) forms a positive definite quadratic function, allowing the deployment of specialized machinery such as the Conjugate Gradient (CG) method. CG has been successfully deployed in some DCF tracking approaches [7, 8, 34]. While a full description of CG is outside the scope of this paper (see [33] for a full treatment), intuitively it finds an optimal search direction \( p \) and step length \( \alpha \) in each iteration. Since the CG algorithm consists of simple vector operations, it can be implemented with only a few lines of python code. The only challenging aspect of CG is the evaluation of the operation \( J^T_w J_w p \) for a search direction \( p \).

We utilize the backpropagation functionality of modern deep learning frameworks, such as PyTorch, for evaluating \( J^T_w J_w p \). Consider a vector \( u \) of the same size as the residuals \( r(w) \). By computing the gradient of their inner product, we obtain \( \frac{\partial}{\partial w} (r(w)^T u) = \frac{\partial r}{\partial w}^T u = J^T_w u \). In fact, this is the standard operation of the backpropagation procedure, namely to apply the transposed Jacobian at each node in the computational graph, starting at the output. We can thus define backpropagation of a scalar function \( s \) with respect to a variable \( v \) as \( \text{BackProp}(s, v) = \frac{\partial s}{\partial v} \). Now, as shown above, we have \( \text{BackProp}(r^T u, w) = J^T_w u \). However, this only accounts for the second product in \( J^T_w J_w p \). We first have to compute \( J_w p \), which involves the application of the Jacobian itself (not its transpose). Thankfully, the Jacobian of the function \( u \mapsto J^T_w u \) is trivially \( J^T_w \), since the function is linear. We can therefore transpose it by applying backpropagation. By letting \( h := J^T_w u = \text{BackProp}(r^T u, w) \), we get \( J_w p = \frac{\partial}{\partial u} (h^T p) = \text{BackProp}(h^T p, u) \).

Given the above mentioned result, we outline the entire optimization procedure in algorithm 1. It applies \( N_{GN} \) Gauss-Newton iterations, each encompassing \( N_{CG} \) Conjugate Gradient iterations for minimizing the resulting subproblem (3.3). Each CG iteration requires two BackProp calls for evaluating \( q_1 = J_w p \) and \( q_2 = J^T_w q_1 \), respectively. There is a need for computing \( h = J^T_w u \) once in the outer loop. Note that in each call to BackProp in algorithm 1, one of the vectors in the inner product is treated as constant, i.e., gradients are not propagated through it. This is highlighted as comments in algorithm 1 for clarity. Note that the optimization algorithm is virtually parameter free, only the number of iterations need to be set. In comparison to gradient descent, the CG-based method...
Algorithm 1 Classification component optimization.

Input: Net weights $w$, residual function $r(w)$, $N_{GN}$, $N_{CG}$

1: for $i = 1, \ldots, N_{GN}$ do
2: $r \leftarrow r(w), \ u \leftarrow r$
3: $h \leftarrow \text{BackProp}(r^Tu, w)$  \# Treat $u$ as constant
4: $g \leftarrow -h, \ p \leftarrow 0, \ p_1 \leftarrow 1, \ \Delta w \leftarrow 0$
5: for $n = 1, \ldots, N_{CG}$ do
6: $p_2 \leftarrow p_1, \ p_1 \leftarrow g^Tg, \ \beta \leftarrow \frac{p_1}{p_2}$
7: $p \leftarrow g + \beta p$
8: $q_1 \leftarrow \text{BackProp}(h^Tp, u)$  \# Treat $p$ as constant
9: $q_2 \leftarrow \text{BackProp}(r^Tq_1, w)$  \# Treat $q_1$ as constant
10: $\alpha \leftarrow \frac{p_1}{q_2p}$
11: $g \leftarrow g - \alpha q_2$
12: $\Delta w \leftarrow \Delta w + \alpha p$
13: end for
14: $w \leftarrow w + \Delta w$
15: end for

adaptively computes the learning rate $\alpha$ and momentum $\beta$ in each iteration. Observe that $g$ is the negative gradient of (3.3).

3.3 Implementation Details

Here, we provide the important implementation details for the target classifier.

Classification Model: The first layer in our classification head (3.1) consists of a $1 \times 1$ convolutional layer $w_1$, which reduces the feature dimensionality to 64. As in [8], the purpose of this layer is to limit memory and computational requirements. The second layer employs a $4 \times 4$ kernel $w_2$ with a single output channel. We set $\phi_1$ to identity since we did not observe any benefit of using a non-linearity at this layer. We use a continuously differentiable parametric exponential linear unit (PELU) [36] as output activation: $\phi_2(t) = t, t \geq 0$ and $\phi_2(t) = \alpha(e^{\frac{t}{\alpha}} - 1), t \leq 0$. Setting $\alpha = 0.05$ allows us to ignore easy negative examples in the loss (3.2). We found the continuous differentiability of $\phi_2$ to be advantageous for optimization.

Feature Extraction: We use ResNet-18 pretrained on ImageNet as our backbone network for feature extraction. The features output from block 4 of ResNet-18 are used as input to the classification model. Features are always extracted from patches of size $288 \times 288$ from image regions corresponding to 5 times the estimated target size.

Online Learning: In the first frame, we perform data augmentation by applying varying degrees of translation, rotation, blur, and dropout, similar to [3], resulting in 30 initial training samples $x_j$. We then apply algorithm 1 with $N_{GN} = 6$ and $N_{CG} = 10$ to optimize the parameters $w$. Subsequently, we only optimize the final layer $w_2$, using $N_{GN} = 1$ and
$N_{CG} = 5$ every 10th frame. In every frame, we add the extracted feature map $x_j$ as a training sample, annotated by a Gaussian $y_j$ centered at the estimated target location. The weights $\gamma_j$ in (3.2) are updated with a learning rate of 0.01.

**Hard Negative Mining:** To further robustify our classification component in the presence of distractors, we adopt a hard negative mining strategy, common in many visual trackers [28, 45]. If a distractor peak is detected in the classification scores, we double the learning rate of this training sample and instantly run a round of optimization with standard settings ($N_{GN} = 1$, $N_{CG} = 5$). This is done so that the model can quickly adapt to the distractors in the scene. We also determine the target as lost if the score falls below 0.25. In such a case, we do not update the target state, remaining "locked" to the previous known state instead.
We describe the proposed target estimation component in this chapter. The aim of our target estimation component is to determine the object bounding box given a rough initial estimate. Our approach is based on the IoU-Net [16], which was recently proposed for object detection as an alternative to typical anchor-based bounding box regression techniques. In contrast to conventional approaches, the IoU-Net is trained to predict the IoU overlap between an image object and an input bounding box candidate. Bounding box estimation is then performed by maximizing the IoU prediction.

The rest of the chapter is organized as follows. Section 4.1 provides a brief overview of the target estimation module. The network architecture used for IoU prediction is described in section 4.2. The different approaches investigated for utilizing the target reference information for IoU prediction are described in section 4.3. Section 4.4 describes the strategy used for training the IoU predictor. Section 4.5 explains how the target estimation module is used for online tracking.

4.1 Overview

In our approach, target estimation is performed by using an IoU-predictor network. This network is trained offline on large-scale video tracking and object detection datasets, and its weights are frozen during online tracking. The IoU-predictor takes four inputs: i) backbone features from current frame, ii) bounding box estimates in the current frame, iii) backbone features from a reference frame, iv) the target bounding box in the reference frame. It then outputs the predicted Intersection over Union (IoU) score for each of the current-frame bounding box estimates. During tracking, the final bounding box is obtained by maximizing the IoU score using gradient ascent. We use block 3 and block 4 features from ResNet-18 as input to our target estimation module. However, the estimation module can be easily modified to use features from any other network architectures. Figure 4.1 provides an overview of our target estimation component.
4.2 Network Architecture

For the task of object detection, independent IoU-Nets are trained in [16] for each object class. However, in tracking the target class is generally unknown. Further, unlike object detection, the target is not required to belong to any set of pre-defined classes or be represented in any existing training datasets. Class-specific IoU predictors are thus of little use for generic visual tracking. Instead, target-specific IoU predictions are required, by exploiting the target annotation in the first frame.

One possible way to get target specific IoU prediction in tracking would be to train an IoU-Net online on the first frame. This is however infeasible as IoU prediction is a high-level task requiring large amounts of training data. In fact, for the object detection task, IoU-Nets are trained on datasets containing over 100K images. Another option is to train a generic, class independent IoU-Net offline, and fine-tune it online using the first frame. This approach is also sub-optimal as the network will be overfitted to a single pose of the target object in the first frame. Thus, the target estimation network needs to be trained offline to learn a general representation for IoU prediction. However, it should also be able to use the target annotation in the first frame to provide target-specific IoU predictions.

Our target estimation network consists of two branches, namely reference branch and test branch, followed by a predictor. Both the branches take backbone features from ResNet-18 Block3 and Block4 as input (see figure 4.2). The reference branch takes the features from the first frame, $x_0$, along with the target annotation $B_0$ as input and...
4.2 Network Architecture

Figure 4.2: The network consists of two branches, denoted reference and test. The reference branch takes an input image of the target, along with ground truth bounding box and provides one-dimensional vector which contains reference information. The test branch outputs a feature representation for each candidate bounding box for the test frame. These are then input to a predictor which estimates IoU for each candidate box.

outputs a target representation. The input features are passed through a convolutional layer. The regions defined by the target bounding box $B_0$ in the resulting feature map are then pooled to a fixed size using a precise pooling (PrPool) layer. Note that the Block4 features are pooled to a $1 \times 1$ grid, while Block3 features are pooled to $3 \times 3$ grid as they have a higher resolution. The pooled features from Block3 are passed through a fully connected layer to get an output of size $1 \times 1$. This is then concatenated with the pooled features from Block4 to get a 1-dimensional reference vector $r(x_0, B_0)$ of size $1 \times 1 \times D_r$. The reference vector contains the target information from the reference image.

The test branch on the other hand takes as input the features $x$ current frame for which the bounding-box needs to be predicted, along with an approximate bounding box estimate $B$. The input features are passed through two convolutional layers to get a general representation suited for IoU prediction. Similar to the reference branch, the region defined by the bounding box estimate $B$ is then pooled to a fixed size using a PrPool. Since these pooled features are used to make the IoU prediction for the test frame, it is desirable to have them in higher resolution. Thus a higher grid size is used for PrPool in the test branch ($3 \times 3$ and $5 \times 5$ for Block3 and Block4 respectively), as compared to that in the reference branch. We denote these pooled features, which are the output of the test branch, as $t(x, B) = t_3(x, B), t_4(x, B)$. Here, $t_3(x, B)$ are the pooled features from Block3 of size $5 \times 5 \times D_t$, while $t_4(x, B)$ are the pooled features from Block4 of size $3 \times 3 \times D_t$. 
The predictor module $g$ takes the reference branch output $(r(x_0, B_0))$ and the test branch output $t(x, B)$ and predicts the IoU between the target and the bounding box estimate in the test frame. The predicted IoU of the bounding box estimate $B$ is hence given by

$$\text{IoU}(B) = g(r(x_0, B_0), t(x, B)).$$ \hspace{1cm} (4.1)

From (4.1) it is clear that the entire target estimation network can be trained offline in an end-to-end fashion, using bounding-box-annotated image pairs. The network can learn to make use of the target appearance in the reference branch and provide target specific IoU predictions. Thus, during online tracking, we can exploit the target information in the first frame by just using it as an input to the reference branch, without the need for any fine-tuning.

The key step in the target estimation network is to use reference frame information $r(x_0, B_0)$ to modify the general representation $t(x, B)$ obtained from the test branch to a target-specific representation, which can then be used by the predictor to make target-specific IoU predictions. We investigate three different approaches to perform this fusion of reference branch and test branch outputs. These are described in the next section.

### 4.3 Reference Fusion

We investigate three different approaches to fuse the target information $r(x_0, B_0)$ with the test branch representation $t(x, B)$, a concatenation based approach, siamese approach and a modulation based approach.

**Concatenation:** In this approach, we use a fully connected layer after the test branch to convert the test branch output $t(x, B)$ to a 1-dimensional vector. This vector is then concatenated with the reference frame information $r(x_0, B_0)$. The concatenated output is then passed through another fully connected layer which makes the IoU prediction. The network architecture for this approach is illustrated in figure 4.3a.

**Siamese:** Similar to the concatenation approach, the output of the test branch is first passed through a fully connected layer to get a 1-dimensional vector with the same number of elements as in $r(x_0, B_0)$. The final IoU prediction is then obtained as a scalar product of this concatenated vector and $r(x_0, B_0)$. See figure 4.3b for an illustration.

**Modulation:** In this approach, the output from the reference branch is passed through two separate fully connected layers to get two modulation vectors $c_3(r(x_0, B_0))$ and $c_4(r(x_0, B_0))$ of size $1 \times 1 \times D_t$ each. The modulation vectors consist of only positive coefficients, and encodes which features channels are most useful for making IoU prediction for the current target. The output from the test branch, $t_3(x, B)$ and $t_4(x, B)$ are then modulated using the vectors $c_3(r(x_0, B_0))$ and $c_4(r(x_0, B_0))$, respectively, via channel-wise multiplication. This creates a target-specific representation for IoU prediction, effectively incorporating the reference appearance information. The modulated representations from the two blocks are then passed through a fully connected layer and
concatenated. This is then passed through another fully connected layer which predicts the IoU (see figure 4.3c).

4.4 Training

We train the target estimation network to minimize the prediction error of (4.1), given annotated data. We use the training splits of the recently introduced Large-scale Single Object Tracking (LaSOT) dataset [9], TrackingNet [27] and the COCO dataset [23]. LaSOT consists of 1400 tracking video sequences, each of which contains 2512 frames on average. The target in each sequence is from one of the 70 classes, and each class contains 20 sequences. As per protocol, we use 18 sequences from each class (1260 in total) for training. TrackingNet is a large scale video dataset consisting of videos sampled from YouTube. The training split is divided into 11 sets, each containing 2511 sequences. We
use only the first 4 sets for training. In contrast to LaSOT, TrackingNet consists of targets belonging to only 21 classes. COCO is an object detection dataset consisting of 80 classes. Similar to [45], we generate synthetic video sequences from COCO by applying random translations and scaling to the images. The reason for using the COCO dataset is to introduce more diverse classes in the training data.

In our framework, each training sample consists of an image pair sampled from a sequence. To generate a training sample, we first select one of the three datasets with equal probability. A sequence is then uniformly sampled from the selected dataset. Finally, image pairs are sampled from the sequence, with a maximum gap of 100 frames. From the reference image, we sample a square patch centered at the target, with an area of about $5^2$ times the target area. From the test image, we sample a similar patch, with some perturbation in the position and scale to simulate the tracking scenario. These cropped regions are then resized to a fixed size.

For each image pair we generate 16 candidate bounding boxes. Ideally, we want to generate candidates whose distribution is similar to the candidate distribution encountered in online tracking. However this distribution is difficult to model. We generate the candidates by adding Gaussian noise to the ground truth coordinates. The variance for the Gaussian are a list of values. If a low variance is sampled, we will get candidates which have high overlap with the groundtruth. A high variance on the other hand will generate candidates which are far off from the groundtruth. This models the scenario where the tracker has drifted from the target, which results in noisy candidates. When generating the candidates, we ensure a minimum IoU of 0.1 with the groundtruth. This is done so that the network can focus on performing accurate target estimation, instead of also performing target classification. We use image flipping and color jittering for data augmentation. As in [16], the IoU is normalized to $[-1, 1]$.

The weights in our target estimation network are initialized using [13]. For the backbone network ResNet-18, we freeze all weights during training. We use the mean-squared error loss function and train for 40 epochs with 40,000 image pairs per epoch. The batch size used is 64. The ADAM [18] optimizer is employed with an initial learning rate of 0.01. The learning rate is reduced by a factor 0.2 every 15 epochs.

### 4.5 Online Tracking

During online tracking, the target reference information $r(x_0, B_0)$ is computed by passing the first frame, along with the annotation, through the reference branch. While it is possible to update this representation by using the last tracked frame for which we have determined the target state, we refrain from doing this since it is possible that the previously estimated state was incorrect.

For each frame, we first extract features at the previously estimated target location and scale. We then apply the classification model (3.1) and find the 2D-position with the maximum confidence score. This provides a rough estimate for the target location in the current frame. Together with the previously estimated target width and height, this generates the initial bounding box $\hat{B}$. As the target size is not expected to change substantially between two consecutive frame, the estimates of target width and height from the previous frame can serve as a good candidate for the target state in the current frame.
The target estimation network is then used to predict the IoU (4.1) for the candidate box $B$. The predicted IoU is then maximized w.r.t. $B$ to get a more accurate estimate of the target bounding box. This is done by performing a fixed number ($N_{GA}$) of gradient ascent iterations with a step length of 1. Similar to [16], the optimization is performed in the log coordinates, i.e. the bounding box $B$ is parametrized as $B = (c_x/w, c_y/h, \log w, \log h)$, where $(c_x, c_y)$ are the image coordinates of the bounding box center, $w$ and $h$ are the target width and height, respectively.

While it is possible to perform state estimation using a single candidate $B$, there is a risk that the gradient ascent can get stuck at a local maxima. To avoid this, we generate a set of $N_{BB}$ initial proposals by adding uniform random noise to the base candidate $B$. The final prediction is obtained by taking the mean of the $K$ bounding boxes with highest IoU. We do not perform any further post-processing or filtering as in e.g. [20] to enforce temporal smoothness in the bounding box coordinates. The final bounding box prediction is used to annotate the training sample $(x_j, y_j)$, as described earlier (section 3.3). They are also used to generate the bounding box candidate $B$ in the next frame.
5

Experiments

5.1 Evaluation Methodology

We evaluate our approach on four standard tracking benchmarks: Need for Speed (NFS) [10], UAV123 [26], TrackingNet [27], and VOT2018 [19]. Note that there is no overlap between the videos used for training the target estimation component and the evaluation datasets. The datasets and the evaluation metrics are described in detail in the next sections. There are two standard protocols to evaluate trackers. NFS, UAV123 and TrackingNet use one pass evaluation (OPE) protocol. Here, the tracker is first initialized with the groundtruth bounding box in the first frame. The tracker is then run on all subsequent frames and the tracking performance is evaluated by comparing the tracker predictions with the groundtruth in every frame. In this approach, a tracker which fails, i.e. loses track of target in the initial frames will be penalized heavily, as all subsequent predictions will be incorrect. To avoid this, the VOT2018 protocol compares the tracker prediction with the groundtruth in every frame while the tracker is running. If the IoU overlap between the tracker prediction and groundtruth is zero, the tracker is declared to have failed. In this case, VOT2018 toolkit skips 5 frames and then re-initializes the tracker with the groundtruth. The performance of the tracker is computed based on the number of tracking failures as well as the overlap between the tracker prediction and the groundtruth for all the frames.

Note that due to the randomness in the candidate generation process in the target estimation (section 4.5), our final tracker is stochastic in nature. Thus, to get a robust estimate of the tracker performance, we run the tracker multiple times over each sequence and report the average performance. On NFS and UAV123, we report the average over 5 runs. On VOT18, as per the standard protocol for stochastic trackers, we report the average over 15 runs. TrackingNet uses an online evaluation server where results over only a single run can be computed. Thus we do not perform multiple runs on TrackingNet.
5.1.1 Evaluation Metrics

Here, we describe the various metrics used to evaluate the trackers.

**Overlap Precision:** For some IoU threshold $T$, overlap precision metric ($OP_T$) [40], is defined as the percentage of frames for which the IoU overlap between the bounding box predicted by the tracker and the groundtruth is larger than the threshold $T$.

**Success Plot:** The success plot [40] is obtained by plotting the overlap precision $OP_T$ for all values of threshold $T$ in the range $[0, 1]$. The area under this curve (AUC) is then used to rank different trackers.

**Distance Precision:** Distance precision (DP) [40] is computed as the percentage of frames for which the distance between the predicted target center and the groundtruth is less than some threshold $D$. For this work, the value of threshold $D$ will be set to 20 pixels. Note that distance precision only uses the predicted target center for evaluation. The predicted target width and height are not utilized for evaluating the tracker. This is in contrast to $OP_T$, where the complete bounding box prediction is used. Thus $OP_T$ is regarded as a better metric to evaluate trackers, as compared to DP.

**Normalized Precision:** The DP metric is sensitive to the target size. If the target is small, then even a small absolute error in the center prediction is significant. However for a large object, an error of a few pixels in center prediction can be considered negligible. Normalized precision [27] aims to address this problem. Instead of considering the absolute error in pixels, normalized precision computes errors relative to the target size. This relative errors are then plotted for the range $0$ to $0.5$. The area under this curve is called Normalized Precision, and is used to rank trackers.

**Robustness:** In the VOT18 evaluation protocol, if the IoU overlap between the prediction and groundtruth is zero, a tracking failure is reported. The tracker will then be re-initialized after 5 frames. Robustness is a value proportional to the number of tracking failures in a dataset. A low robustness value indicates that the tracker is less likely to fail on a sequence.

**Accuracy:** Accuracy is another evaluation metric used by VOT18. It is computed as the mean IoU overlap between the tracker prediction and the groundtruth during successful tracking periods. When calculating accuracy, 10 frames after a re-initialization are ignored to reduce bias as the IoU overlap is likely to be high for a few frames after an initialization.

**EAO:** Expected Average Overlap (EAO) is a metric used by VOT18 to rank the trackers. EAO combines both the accuracy and robustness measures to provide a single score. EAO denotes the average overlap between the predictions and the groundtruth which a tracker is expected to have when the average is computed over a large number of sequences. We refer to [19] for more details.
5.1 Evaluation Methodology

5.1.2 Datasets

In this section, we describe the datasets which we use to evaluate the trackers.

NFS: The Need For Speed benchmark [10] consists of 100 video captured from high frame rate (240 FPS) cameras, as well as their 30 FPS versions. The target bounding box in every frame is manually annotated. Further, all sequences are labelled with nine visual attributes such as occlusion, fast motion, which indicate the main tracking challenges in the particular sequence. The benchmark uses AUC and $OP_T$ metrics to rank the trackers.

UAV123: UAV123 benchmark [26] consists of 123 aerial videos captured from an unmanned aerial vehicle (UAV). A few sequences (8) in the dataset are synthetically generated using a simulator. As the videos are captured from a moving platform, changes in target pose due to camera motion is extremely common in this dataset. Similar to NFS, the target is annotated in each video and the sequences are labelled with twelve attributes. Trackers are ranked using the AUC measure.

TrackingNet: TrackingNet is a recently introduced large scale dataset for object tracking. It consists of a training split of 30132 videos which can be used to train trackers, as well as a test split of 511 videos used to evaluate the trackers. The videos are collected from Youtube-BoundingBoxes [29] which is a large scale video object detection dataset. In contrast to NFS and UAV123 which contain videos specially captured for generating datasets, TrackingNet consists of real-world videos from YouTube and thus can give a better estimate of real-world tracking performance. TrackingNet uses AUC, DP and Normalized Precision metrics to evaluate trackers. The groundtruth annotations for the test set are withheld, and the trackers are evaluated using an online evaluation server.

VOT18: Visual Object Tracking (VOT) is a tracking challenge held every year since 2013. Compared to other benchmarks which use one pass evaluation, VOT performs a supervised evaluation where in case of tracking loss (i.e. zero overlap between the prediction and groundtruth), the tracker is re-initialized after 5 frames. We perform our experiments on the dataset corresponding to the 2018 version of the challenge. This dataset consists of 60 videos carefully selected from a pool of 443 videos. The trackers are evaluated using accuracy, robustness and EAO metrics.

5.1.3 Tracker Settings

Unless explicitly stated otherwise, we use the following tracker settings in all our experiments. We employ the modulation based approach to fuse reference image information for IoU prediction. The number of gradient ascent steps $N_{GA}$ for IoU maximization in each frame is set to 5. We use $N_{BB} = 10$ bounding box candidates in each frame. The final bounding box prediction is obtained as mean of $K = 3$ candidates with highest predicted IoU.
Table 5.1: Analysis of different architectures for fusing the reference image information for IOU prediction. Results are provided on the combined NFS and UAV123 datasets. The baseline approach, which does not employ a reference branch to integrate target specific information, provides poor results. Among the different architectures, the modulation based approach achieves the best results.

<table>
<thead>
<tr>
<th></th>
<th>Baseline</th>
<th>Modulation</th>
<th>Concatenation</th>
<th>Siamese</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP_{0.50}(%)</td>
<td>68.3</td>
<td><strong>76.3</strong></td>
<td>67.5</td>
<td>75.1</td>
</tr>
<tr>
<td>OP_{0.75}(%)</td>
<td>38.6</td>
<td><strong>48.4</strong></td>
<td>37.9</td>
<td>47.6</td>
</tr>
<tr>
<td>AUC (%)</td>
<td>56.7</td>
<td><strong>62.3</strong></td>
<td>56.3</td>
<td>61.7</td>
</tr>
</tbody>
</table>

5.2 Baseline Experiments

In this section, we perform baseline experiments to evaluate the approaches proposed in the thesis. The experiments are performed on the combined NFS and UAV123 datasets, consisting of 223 videos. We will refer to this combined set as NFS+UAV123. We use OP_{0.5}, OP_{0.75} and AUC score to evaluate the trackers. The three network architectures for reference image fusion are compared in section 5.2.1. Section 5.2.2 investigates the impact of using different ResNet-18 features blocks for IOU prediction. The impact of using multiple proposals for target estimation is investigated in section 5.2.3. Section 5.2.4 evaluates how the tracking performance is affected as the number of gradient ascent iterations performed in IOU maximization is varied. Section 5.2.5 compares the proposed target estimation approach with the traditional multi-scale search approach.

5.2.1 Reference Fusion

Here, we compare the different approaches discussed in section 4.3 for fusing the reference image information for IOU prediction. We also evaluate a baseline approach which does not use the reference image information. That is, the baseline network only uses the test frame to predict the IOU. The results are shown in table 5.1. We observe that the modulation approach obtains the best results in all metrics, with an AUC score of 62.3%. The Siamese approach for reference fusion also obtains competitive results, with AUC score of 61.7% and OP_{0.75} score of 47.6. The concatenation approach on the other hand obtains poor results which are slightly worse than the baseline approach using only the test frame. This indicates that the concatenation approach fails to utilize the reference image information. The baseline approach which does not use the reference image obtains significantly poor results, as compared to the Modulation or Siamese approach. This demonstrates the importance of exploiting target-specific appearance information in order to accurately predict the IOU for an arbitrary object.
5.2 Baseline Experiments

### Table 5.2: Analysis of using different ResNet-18 feature blocks as input for IoU prediction. Results are provided on the combined NFS and UAV123 datasets. Block3 which contains high resolution, lower level semantic information as compared to Block4 provides better results. The best results are obtained by using both blocks as input, indicating that they contain complementary information for IoU prediction.

<table>
<thead>
<tr>
<th></th>
<th>Block 3&amp;4</th>
<th>Block 3</th>
<th>Block 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP(_{0.50}) (%)</td>
<td>76.3</td>
<td>73.4</td>
<td>73.6</td>
</tr>
<tr>
<td>OP(_{0.75}) (%)</td>
<td>48.4</td>
<td>44.5</td>
<td>38.9</td>
</tr>
<tr>
<td>AUC (%)</td>
<td>62.3</td>
<td>60.3</td>
<td>58.5</td>
</tr>
</tbody>
</table>

### Table 5.3: Analysis of using multiple initial proposals for target state estimation. Results are provided on the combined NFS and UAV123 datasets. Using a single proposal provides poor results, as compared to using multiple proposals (>= 5). The best results, in terms of AUC score, are obtained by using 10 proposals.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>5</th>
<th>10</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP(_{0.50}) (%)</td>
<td>74.1</td>
<td>76.3</td>
<td><strong>76.3</strong></td>
<td>76.2</td>
</tr>
<tr>
<td>OP(_{0.75}) (%)</td>
<td>46.5</td>
<td>47.8</td>
<td>48.4</td>
<td><strong>48.7</strong></td>
</tr>
<tr>
<td>AUC (%)</td>
<td>61.2</td>
<td>62.2</td>
<td><strong>62.3</strong></td>
<td>62.2</td>
</tr>
</tbody>
</table>

5.2.2 Backbone features layers

We evaluate the impact of using different feature blocks from the backbone ResNet-18 (table 5.2). Using features from only Block3 leads to an AUC of 60.3%, while only Block4 gives an AUC of 58.5%. Fusing features from both the blocks leads to a significant improvement, giving an AUC score of 62.3%. This indicates that Block3 and Block4 features have complementary information useful for predicting the IoU.

5.2.3 Number of proposals

Here, we investigate the impact of using multiple initial proposals (\(N_{BB} > 1\)) for target state estimation (section 4.5). Table 5.3 shows the tracking performance for different values of \(N_{BB}\). We observe that using multiple initial proposals is indeed beneficial and provides \(\approx 1\%\) improvement is AUC score, as compared to using a single initial proposal. The best results are obtained with \(N_{BB} = 10\), although \(N_{BB} = 5\) also gives similar results. Also note that using an even higher number of proposals (\(N_{BB} > 10\)) does not lead to any further improvement, indicating that using only a few initial proposals is sufficient to avoid local maxima.
Experiments

5.2.4 Number of iterations

We investigate the impact of the number of gradient ascent iterations, $N_{GA}$ (section 4.5), on tracking performance. The results are show in table 5.4. We observe that using a higher number of iterations lead to better tracking performance. The best results are obtained using $N_{GA} = 10$. We did not experiment with using even more number of gradient ascent iterations ($N_{GA} > 10$) due to lack of time. For our final tracker, we used $N_{GA} = 5$ to obtain higher tracking speed with competitive tracking performance.

5.2.5 Multi-Scale search

We compare our target state estimation component with a brute-force multi-scale search approach employing only the classification model. This approach mimics the common practice in correlation filter based methods. We extract features at 5 different scales with a scale ratio of 1.02. The classification component is then evaluated on all scales, selecting the location and scale with the highest confidence score as the new target state. Results are shown in table 5.5. Our approach significantly outperforms the multi-scale method by 8.6% in AUC. Further, our approach almost doubles the percentage of highly accurate bounding box predictions, as measured by OP$_{0.75}$. These results highlight the importance of treating target state estimation as a high-level visual task.

### Table 5.4: Analysis of using different number of gradient ascent iterations for target state estimation. Results are shown on the combined NFS and UAV123 datasets. Using a higher number of gradient ascent iterations leads to better performance and the best results are obtained using 10 gradient ascent iterations.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP$_{0.50}$ (%)</td>
<td>75.3</td>
<td>76.0</td>
<td>76.3</td>
<td><strong>77.1</strong></td>
</tr>
<tr>
<td>OP$_{0.75}$ (%)</td>
<td>45.6</td>
<td>48.4</td>
<td>48.4</td>
<td><strong>48.6</strong></td>
</tr>
<tr>
<td>AUC (%)</td>
<td>61.1</td>
<td>62.9</td>
<td>62.3</td>
<td><strong>62.7</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Ours</th>
<th>Multi-Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>OP$_{0.50}$ (%)</td>
<td><strong>76.3</strong></td>
<td>66.2</td>
</tr>
<tr>
<td>OP$_{0.75}$ (%)</td>
<td><strong>48.4</strong></td>
<td>26.0</td>
</tr>
<tr>
<td>AUC (%)</td>
<td><strong>62.3</strong></td>
<td>53.7</td>
</tr>
</tbody>
</table>

### Table 5.5: Comparison of the proposed target estimation approach with the brute-force multi-scale approach commonly employed by state-of-the-art trackers. Results are provided on the combined NFS and UAV123 datasets. The proposed approach based on IoU prediction obtains significantly better results.
5.3 State-of-the-art Evaluation

In this section, we compare our tracker, named ATOM with the state-of-the-art approaches on NFS, UAV123, TrackingNet and VOT18 benchmarks.

5.3.1 Need For Speed

We compare our approach with UPDT, CCOT, ECO, MDNet, HDT, DaSiamRPN, FCNT, SRDCF and BACF. Figure 5.1 shows the success plot over all the 100 videos, reporting AUC scores in the legend. CCOT and UPDT, both based on correlation filters, achieve AUC scores of 49.2% and 54.2% respectively. Our tracker significantly outperforms UPDT with a relative gain of 9%.

5.3.2 UAV123

We compare our approach with DaSiamRPN, SiamRPN, UPDT, ECO, CCOT, SRDCF, Staple, ASLA and SAMF. Figure 5.2 displays the success plot over all the 123 videos. DaSiamRPN and its predecessor SiamRPN employ a target estimation component based on bounding box regression. Compared to other approaches, DaSiamRPN achieves a superior AUC of 58.4%, owing to its accuracy. Our tracker, employing an overlap maximization strategy for target estimation, significantly outperforms DaSiamRPN by achieving an
Experiments

5.3.3 TrackingNet

We compare our approach with UPDT, MDNet, CFNet, SiameseFC, DaSiamRPN, CSRDCF, SAMF and Staple. Table 5.6 shows the results in terms of precision, normalized precision, and success. In terms of precision and success, MDNet achieves scores of 56.5\% and 60.6\% respectively. Our tracker outperforms MDNet by achieving relative gains of 14\% and 16\% in terms of precision and success respectively.

5.3.4 VOT2018

We compare our approach with the top-10 methods on the VOT18 challenge, LADCF, MFT, DaSiamRPN, UPDT, RCO, DRT, DeepSTRCF, CPT, SASiamR and DLSTpp. Table 5.7 shows the results presented in terms of EAO, robustness, and accuracy. Among the top trackers, only DaSiamRPN uses an explicit target state estimation component, achieving higher accuracy compared to its DCF-based counterparts LADCF, MFT, UPDT, and RCO. Our approach ATOM achieves the best accuracy, while having competitive robustness. Further, our tracker obtains the best EAO score of 0.401, with a relative gain of 3\% over the VOT2018 competition winner.
5.4 Attribute-based Comparison

Here, we perform an attribute analysis on UA V123 dataset. In UA V123, each video is annotated with 12 different attributes: aspect ratio change, background clutter, camera motion, fast motion, full occlusion, illumination variation, low resolution, out-of-view, partial occlusion, scale variation, similar objects, and viewpoint change. Figure 5.3 shows the success plots for all the attributes. Our approach obtains the best results on all 12 attributes. Our approach especially excels in case of aspect ratio change, scale variation, and viewpoint change, demonstrating the impact of our target estimation component.

5.5 Qualitative Results

We show some qualitative results of our tracker on standard tracking benchmark sequences. Figure 5.4 shows the results on two sequences each from UA V123 and NFS datasets. The sequences contain frequent changes in target pose leading to large changes in the target state, and thus are suited to evaluate our approach. For comparison, we include the outputs of two other state-of-the-art trackers UPDT [3] and DaSiamRPN [45]. UPDT employs multi-scale search for determining the target scale, while DaSiamRPN employs the bounding box regression technique. The results show that our approach ATOM, employing an overlap prediction network, provides much accurate bounding box predictions, as compared to UPDT and DaSiamRPN.
Figure 5.3: Attribute analysis on the UAV123 dataset. Our approach ATOM obtains the best performance on all 12 attributes.
5.5 Qualitative Results

Figure 5.4: A comparison of our approach with state-of-the-art trackers. UPDT [3], based on correlation filters, lacks an explicit target state estimation component, performing a brute-force multi-scale search instead. Consequently, it does not handle aspect-ratio changes, which can lead to tracking failure (second row). DaSiamRPN [45] employs a bounding box regression strategy to estimate the target state, but still struggles in cases of out-of-plane rotation, deformation, etc. Our approach ATOM, employing an overlap prediction network, successfully handles these challenges and provides accurate bounding box predictions.
Conclusion

In this thesis, the problem of accurate tracking was studied. A target estimation component to predict accurate bounding box using overlap maximization was proposed. Different architectures to exploit the reference frame information for target specific IoU overlap were developed and compared, and a modulation based approach was found to provide the best results. The target estimation module was integrated with a robust tracker to obtain robust and accurate tracking.

Extensive baseline experiments were performed on two tracking benchmarks to investigate the impact of the various design choices made in the thesis. Further, the proposed approach was compared with the state-of-the-art tracking approaches on four standard tracking benchmarks. The approach was shown to provide the best results all four datasets, outperforming other state-of-the-art trackers, thus demonstrating the impact of the proposed approach.


